Aufgabenblatt

Multithreading

Verteilte Anwendungen Praktikum
Hochschule Furtwangen
Prof. Dr. Dirk Eisenbiegler

Aufgabe 1 - Eieruhr

In dieser Aufgabe soll eine Eieruhrfunktion realisiert werden: Einmal gestartet gibt eine Eieruhr nach einer vorgegebenen Zeit einen Alarm aus.

Realisieren Sie die Eieruhrfunktion in Form einer statischen Methode mit dem Namen *eieruhr*. Die Methode soll zwei Parameter haben: eine Zeitangabe in Millisekunden und einen Ausgabetext, der zum Alarmzeitpunkt ausgegeben werden soll. Innerhalb der Methode eiseruhr soll ein Thread erzeugt werden, der zunächst die vorgegebene Zeit lang wartet und danach die Ausgabe auf dem Bildschirm erzeugt. Der Thread, der die Methode eieruhr aufruft, soll nach dem Aufruf sofort weiterarbeiten können.

Tipps:

- ✗ Erzeugen und starten Sie innerhalb der eieruhr-Methode einen neuen Thread. Dazu müssen Sie zunächst eine Sohnklasse von Thread implementieren und deren run-Methoden geeignet überschreiben.
- ✗ Verwenden Sie folgende Methode, um einen Thread für eine vorgegebene Zeit (m Sekunden) warten zu lassen:

```
public static void schlafen(int m) {
   try {
       Thread.sleep(1000*m);
   } catch (InterruptedException t) {
   }
}
```

A) Implementieren Sie zunächst die Methode eieruhr. Testen Sie die Methode, indem Sie sie mehrfach hintereinander mit unterschiedlichen Parameterwerten aufrufen.

B) Modifizieren Sie die Methode derart, dass die Eieruhr jede Sekunde die noch verbleibende Zeit und die Nachricht ausgibt.

Stand 16.09.2020

Aufgabe 2 - Dispatcher

Gegeben folgende Schnittstelle:

```
public interface F {
    public int f (int x);
}
```

Es soll eine statische Methode mit dem Namen *execute* programmiert werden. Die Methode *execute* hat zwei Parameter: einen Parameter *f* vom Typ *F* und einen Parameter *n* vom Typ *int*. Die Methode execute soll die Werte *f(0)*, *f(1)*, *f(2)* ... *f(n-1)* bestimmen, diese Werte in einem int-Array nacheinander ablegen und schließlich diesen int-Array zurückgeben.
Signatur von *execute*:

```
public static int[] execute(F f, int n)
```

Die Methode *execute* soll die Funktionsaufrufe *f(0)*, *f(1)*, *f(2)* ... *f(n-1)* nicht nacheinander ausführen, sondern vielmehr sollen diese Funktionsaufrufe gleichzeitug je durch einen Thread ausgeführt werden.

Tipps:

- ✗ Starten Sie innerhalb von *execute* *n* Threads. Übergeben Sie jedem der Threads eine der *x*-Werte 0, 1, 2, ... *n*-1, sowie das Objekt *f*. Der Thread soll dann *f(x)* bestimmen.
- ✗ Erzeugen Sie in der Methode *execute* zusätzlich ein Objekt mit von der selbstprogrammierten Klasse *Result*. Übergeben Sie auch dieses Objekt jedem der Threads.
- ✗ In das Objekt vom Typ Result sollen die Threads ihre Einzelergebnisse ablegen können und der Hauptthread (*execute*-Methode) soll daraus das Gesamtergebnis auslesen können. Realisieren Sie diese beiden Funktionen je als eine Objektmethode.
- ✗ Beim Auslesen des Gesamtergebnisses soll der Aufrufer solange blockiert werden, bis alle *n* Ergebnisse vorliegen. Kennzeichnen Sie die beiden Methoden von Result als synchronized und verwenden Sie innerhalb der Methoden *wait* und *notify* zur Synchronisation.